51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2012年11月
总第020期

11月编程语言排行榜：语言的浮浮沉沉

动态编程语言遍地开花：浅析Ruby的主流

学会像一个函数式程序员那样思考

developer.51cto.com

目录

2012.11 Contents

51CTO.com
技术成就梦想

出版方：北京无忧创想信息技术有限公司

责任编辑：彭凡　林师授　　封面设计：苍旭

联系方法：pengfan@51cto.com　010-68476606（分机 8058）

出版日期：2012 年 11 月 20 日

欢迎广大读者投稿

11月编程语言排行榜:语言的浮浮沉沉

2012 年 11 月 6 日，TIOBE 公布最新一期编程语言排行榜。业界看到 Objective-C 蹿升势头良好，预测它即将蝉联年度编程语言排行榜的宝座。而 C 语言一直压制第二位的 Java，并有差距越来越大的趋势。

2012 年 11 月 6 日，TIOBE 公布最新一期编程语言排行榜。业界看到 Objective-C 蹿升势头良好，预测它即将蝉联年度编程语言排行榜的宝座。而 C 语言一直压制第二位的 Java，并有差距越来越大的趋势。

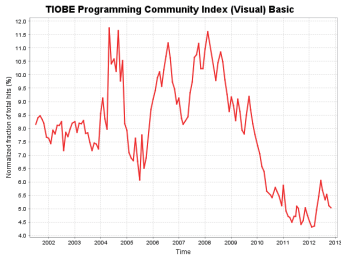
下面是本期编程语言排行榜榜单：

Position Nov 2012	Position Nov 2011	Delta in Position	Programming Language	Ratings Nov 2012	Delta Nov 2011	Status
1	2	↑	C	19.224%	+1.90%	A
2	1	↓	Java	17.455%	-0.42%	A
3	6	↑↑↑	Objective-C	10.383%	+4.40%	A
4	3	↓	C++	9.698%	+1.61%	A
5	5	=	PHP	5.732%	-0.36%	A
6	4	↓↓	C#	5.591%	-1.73%	A
7	7	=	(Visual) Basic	5.032%	-0.01%	A
8	8	=	Python	4.062%	+0.45%	A
9	10	↑	Perl	2.182%	+0.10%	A
10	11	↑	Ruby	1.739%	+0.24%	A
11	9	↓↓	JavaScript	1.278%	-1.29%	A
12	16	↑↑↑↑	Delphi/Object Pascal	0.995%	+0.12%	A
13	13	=	Lisp	0.951%	-0.23%	A
14	14	=	Pascal	0.881%	-0.11%	A
15	23	↑↑↑↑↑↑↑↑	Visual Basic .NET	0.769%	+0.24%	A-
16	19	↑↑↑	Ada	0.662%	+0.04%	B
17	12	↓↓↓↓↓	PL/SQL	0.632%	-0.81%	B
18	18	=	Lua	0.631%	0.00%	A-
19	15	↓↓↓↓	MATLAB	0.620%	-0.34%	B
20	24	↑↑↑↑	Assembly	0.585%	+0.06%	B

51CTO 发布编程语言排行榜已经快 5 年的时光了，在这五年中我们见证了不少语言的起起落落。虽然国内有很多大牛说，关注这语言的排名没有多少意义。但我们从 Objective-C 的上升过程中，还是能看到移动设备端，特别是 iOS 应用开发方面的黄金潜力。

昔日季军(Visual) Basic

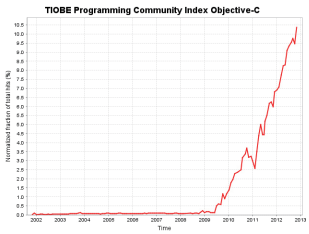
相信很多人都参与过 VB 的开发和学习，Basic 可能是很多程序员学习的第一门语言。在 2008 年 1 月 51CTO 推出的第一期编程语言排行榜中，VB 还排在季军的位置。转眼到了 2012 年年底，它已经悄然跌落到第 7 的位置。



我们可以看到，VB 已经慢慢进入下降通道。昔日红火的语言不在，何时才能翻身呢？

Objective-C 的火箭速度

Objective-C 其实不是一门新语言，1980 年代初布莱德·确斯 (Brad Cox) 在其公司 Stepstone 发明 Objective-C。他对软件设计和编程里的真实可用度问题十分关心。Objective-C 最主要的描述是他 1986 年出版的 Object Oriented Programming: An Evolutionary Approach. Addison Wesley. ISBN 0-201-54834-8。



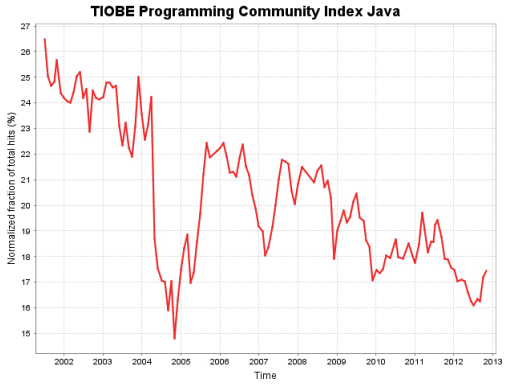
尽管它已经出生近 30 年了，但红火起来却是 iPhone 的诞生。当乔帮主一次次举着 iPhone 对

2012 年 11 月编程语言排行榜：语言的起起伏伏

全世界微笑的时候，想必没多少 iPhone 用户会注意 Objective-C 语言的蹿升速度。在近乎一条 60 度直线的上升曲线中，更多的开发者进入了 iOS 移动应用开发领域。这片领域究竟是蓝海还是即将成为红海，我们拭目以待。

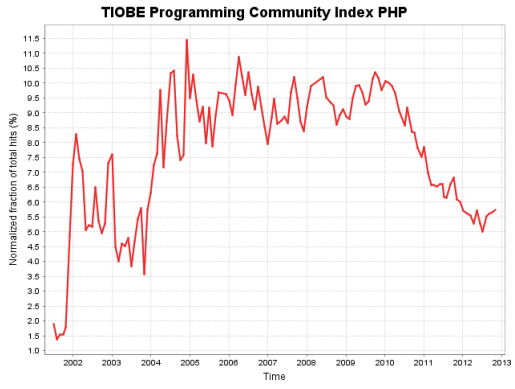
Java 的起起伏伏

看 Java 最近十年的走势，很像中国股市一样。不断的起伏，在起伏中似乎看不到未来的希望。在学校时老师满面春风的告诉我们，Java 跨平台编译的特性是如何如何好。但现在我们确实看到了 Java 的衰落。



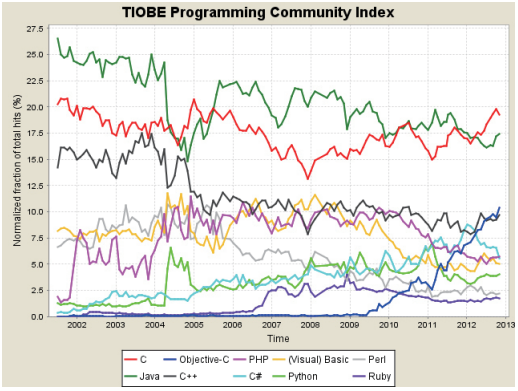
尽管 Android 让 Java 赶上了移动互联网时代，但接下来 Java 还是继续这么衰落下去么？

PHP 预示 WEB 时代的过去？



看 PHP 的火热期与 WEB 互联网大发展同步。LAMP 架构催生了多少新网站和无数的中国小站长。曾几何时，自己架个空间就能做起一个网站来。而 PHP 程序员因此走热。

前 10 名编程语言走势图

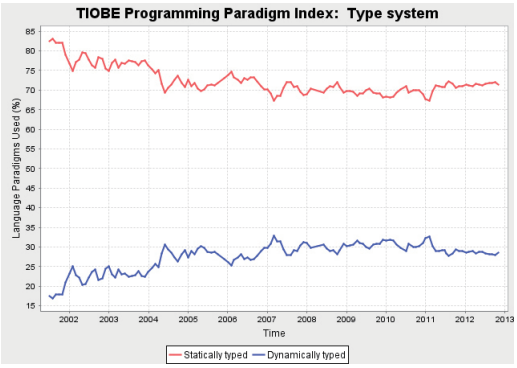


下面是第 50 到 100 的编程语言排名

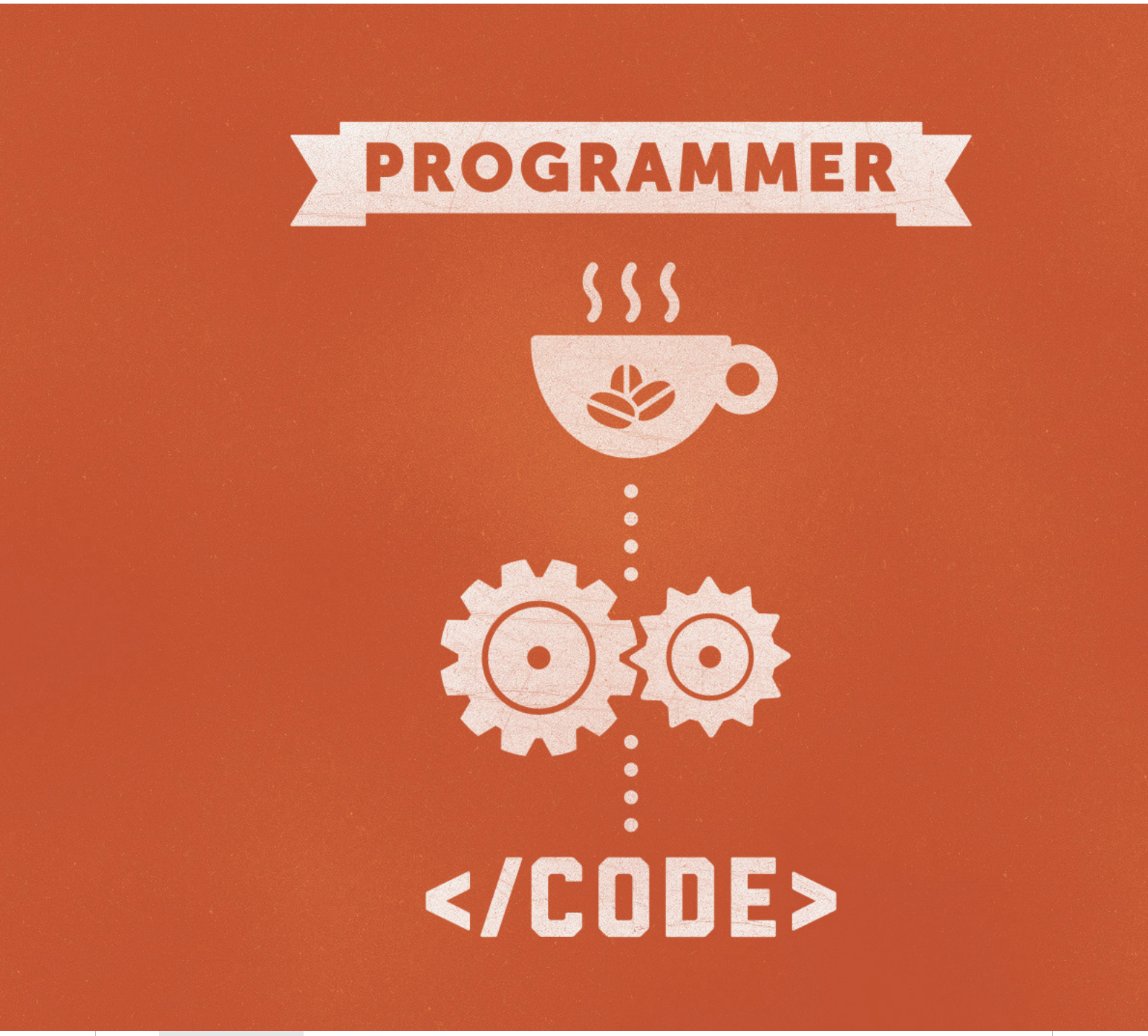
(Visual) FoxPro, ABC, Algol, Apex, AutoLISP, bc, BETA, Boo, C shell, CHILL, CL (OS/400), Clean, Clojure, cT, Dart, Dylan, Emacs Lisp, Euphoria, F#, Factor, Go, Icon, IDL, Inform, Informix-4GL, J, JScript.NET, Ladder Logic, LPC, Mathematica, MUMPS, NATURAL, Oberon, OCaml, Occam, OpenCL, OpenEdge ABL, Oz,

Pike, PL/I, PowerShell, Q, REXX, S, SPARK, VBScript, VHDL, WebDNA, X10, xBase

下面给出了编程语言类别的一年变化趋势



■



2012 年 11 月 11 日，各位程序员你是否找到了自己的另一半？十一月对于程序员来说真的如此残酷？

51CTO 本期专题将为您带来一位阿里巴巴码农的六年开发生涯，给各位单身或者已婚的码农分享一下技术人生。

生命不息！ CODE 不止！

——51CTO 开发频道寄语

一个阿里巴巴码农的六年回眸

本文讲述了一个阿里巴巴码农的六年回眸，它讲述了一个个冷冰冰产品背后的活生生的人了，我们一起来看一下这篇文章，看能从中学到些什么。

本文由淘宝开放平台技术产品负责人 @ 放翁 _ 文初 撰写，它讲述了一个个冷冰冰产品背后的活生生的人了，也在讲述着一个码农的六年心路历程，“技术耐得住寂寞，低谷积累高峰冲刺，主动改变一切。”

10 月 13 日，关于淘宝开放平台技术部分的分享看到有些同学留言说有这样的机会和环境是幸运的，的确在阿里这些年赶上了公司的发展，赶上了互联网技术的发展，是幸运的，但是背后这个普通的人，从进入公司的低级程序员一步一步成长起来到底是怎么走过来的，也许可以让一些正在走的同学有值得思考的地方，我尽量少加一些自己的收获在里面，因为每个人看到这些场景感到了什么那就是什么。

2005 年：走出国企

毕业 4 年多在一家通信国企做的顺风顺水（不是自己能力有多突出，是老板经常被挖，所以提升的不错），总感觉有点养老的味道，于是开始在招聘网站投简历，没想到一家叫阿里巴巴的杭州公司联系上我去面试，聊下来是类似于后台运维部门，兴趣不在此，就拒绝了，但其实埋下了伏笔。

2006 年：一切归零

刚过完年，那家阿里的公司 HR 又联系我，说这次部门不太一样，去了，然后说了一些关于 Work at alibaba 的想法，那时觉得互联网要比通信企业更有意思，所以不顾家人和未来老婆的反对，辞职去了这么一家以前都没听说过的企业。进公司以后，一切归零，工作四年后又从底层做起，虽然忙碌，感觉充实。半年后开始觉得有些无聊，该学的都学到了，剩下的就干一点常规的活，啥 work at alibaba 又没影了，心里落差很大。

2007 年：征召入伍

突然被征召“入伍”，20 来号人被拖到老马（马云）的福地“湖畔花园”创业去了，我的感觉就是莫名其妙，收购了一家公司（主要做 CRM 系统框架和模型驱动），然后就开始要搞创业了，不过起码 work at alibaba 的说法有找落了。一帮子人窝在 3 室一厅的房子里面，不同业务团队分在不同的小屋子里，测试和架构团队在客厅。虽然我是个没有背景的小兵，但老大还是给了我足够的

一个阿里巴巴码农的六年回眸 II

机会，我没有去做业务支撑，反而成了团队里面做基础系统的，这里每个人的 level 都比我高 2 级，于是我背着“架构师”的名字开始努力奋斗。

2008 年：照猫画虎

菲青老大加盟淘宝，成为淘宝的首席架构师，菲青组织集团所有架构团队的同学定期聚到西湖边上的“淘咖啡”（现在已经没了），相互间交流技术，我这毛头小鬼，托福也被拉了进来。翻着 yahoo 的网站，看到了 flickr（记得今年年初的时候还谈当年的 flickr 是最能够成为今天 FB 的公司，结果给雅虎浪费了）的开放模式，于是照猫画虎，还就搞出了一个看起来还有点摸样的东西。接下来我的主业就放在这块上面了，这一年技术成长很快，因为开放平台是新事物，安全（数字签名，授权，加密），数据处理（ xml 解析各种基础知识，json 的规范），REST 代理服务器实现等等，这些都是互联网的产物，都非常扎实落地，充实的日子过的总是很快。

2009 年：阿里软件解体

阿里集团的不同公司的氛围是完全不同的，阿里软件团队氛围还是那种强调自上而下的管理，加上一些关系纠缠在里面，我这个苦脸码农一直都不入流，因此也想换换环境，加上看到当时 HSF 的成长中菲青老大能抗的住压力让毕玄最后坚持下去，心里还是酸酸的，但 HR 和 boss 的左手大棒（你去哪家公司我们管不了你）右手胡萝卜（晋升名额本来就是你的），我能做什么呢（那些日子的经历让我记忆犹新，也许当时运气太好了吧，在淘宝快 3 年多了，每个 HR 都让我觉得受宠若惊）。但世事难料，年中的时候突然宣布公司解体，就这样我们笑话的说我们把公司搞没了，很多人非常伤感，我却非常开心，因为我有机会去淘宝了。

2010 年：空降淘宝

空降淘宝，虽然新老板对我能力比较认可，但是淘宝的开放平台已经有了一个 10 个左右的小团队了，如何融入是最迫切的。我缺乏的是业务，了解的是平台，能力在于技术，于是天天帮助团队同学打杂，解决问题，慢慢的也用能力证明自己。一直处于一个团队攻坚和打杂的角色，技术能力还是得到了飞速的提升，因为这一年开放平台正式商业化了，对于基础平台的要求非常高。但这一年也有些不好的评价对我，有些同学觉得我太强势了，对于团队成员的发展会起到反作用，顺风顺水的我依然觉得用技术说话，判断力取决于技术能力。

2011 年：忐忑生涯

经过一年多的基础平台建设，整体平台架构已经比较完善，而我的角色开始显得有些尴尬，叫架构师，但业务管不着，技术管一块，不带人，干活自己做。后来这年有个毛头小伙子听了我的课死活要加入开放平台（技术大学第一届），因此他成了我第一个徒弟式的同学，然后加上我是淘宝第一个做 App OPS（原来运维是独立于开发的，后来因为希望给开发更多空间，所以允许有开发有能力的人自己管理系统基础运维和发布，这类人被叫做 App OPS），另一个还在实习的愣头小伙子也成了我的徒弟式的同学，就这样我这个架构师有了一点人员资源干点可以干的活。

事情总不按每个人的想象走，最后组织■

本文更详细内容，请查看：

http://developer.51cto.com/art/201210/360763.htm

编者按

假如女人是一种编程语言，也许每个男人心中都有自己最喜欢的那一个吧？我认为女人可分两类，“面向过程”和“面向对象”。

假如女人是一种编程语言

计算机语言的实质其实是为了让人类能够更好与计算机打交道，最终结果都是通过编译成二进制代码或通过解释器转译成二进制代码的形式，由计算机来执行。而二进制就是0和1，假设1代表男人，0代表女人，那么瞬间就可以发现世界就变成了一台大“计算机”。而爱情和情爱也可以看成是1和0组成的二进制机器码。

假如女人是一种编程语言，也许每个男人心中都有自己最喜欢的那一个吧？我认为女人可分两类，“面向过程”和“面向对象”。“面向对象”的女人以寻找终身伴侣为目标，而“面向过程”的，则以经历难以忘怀的爱情时光为己任。各位程序猿当然也要按需选择最适合自己的“编程语言”，不要因为时髦而盲目追求“面向对象“，有时候过程可能比结果更重要。

当然了，“编程”还有多种“范式”（Fashion）可以遵循。各种语言也都有自己独特的“范式”。女人无数种，这里按流行的占星学星座（即不包括蛇夫座）分类，浅表一二。

特别声明：一家之言，仅供娱乐。莫对号入座。

白羊女——Basic 语言

这是一种很简单直白的语言，很多80后程序员的启蒙语言。当然也有很多变种，比如VB的初级白痴，QBasic的清朗直率，Small Basic的小巧精致。和这种“语言”相处，过程很重要，时时保持一个漂亮的“结构化编程”很重要，不要过于追求”面向对象”，最后反而得不偿失。Basic是很容易“编译”成二进制的语言，虽然没有那么强的“底层”需求但也要防止“乱”代码，遗憾的是这只能靠程序猿自觉了。

金牛女——Perl 语言

她想让自己兼有C语言、Shell、awk等等语言的优势，然而用后却发现有点四不像且反而增加了程序猿的压力。同时，这是一种有点自伤的语言，将自由给了程序猿却把伤害深深埋在心底。需要程序猿不断调整“编程”思路，不断更新“编程范式”，给出最佳的解决算法。总而言之，这是一个需要精心照顾的 “语言”，永远不要让这只“骆驼”饥渴！

双子女——Shell 脚本

她很强大，某些时候甚至超过C语言。她也很脆弱，也许一个简单的“rm -fr / usr”（注意那个空格）就能带来毁灭性打击。她可以完成输入输出双向重定向。她可以绝对的“面向过程”，过程的每个细节都要精心推敲，花样百出；同样也可以绝对的“面向对象”，让每一次coding都如家搬温暖，常常一些“变量”更容易让整个“编程”效率倍增。别忘了Makefile的

假如女人是一种编程语言 II

本质也是一个 Shell 脚本，所以增加一个叫做 Love: 的目标吧，然后在命令行下郑重地运行：

```
$ Make Love
```

巨蟹女——LISP 语言

实话实说，这真的是一个又小众又“闷骚”的语言。也许和它的发明者麦卡锡博士的个性有关。说她小众，是因为使用 Lisp 进行开发的人少之又少，很多初学者都会被她敏感的神经吓跑，从 Emacs 的用户量就可以看出 Lisp 只能是有特定品味的人的掌上明珠，而一旦掌握 Lisp 就会发现真的是乐趣无穷而安全感倍增。至于说她“闷骚”，因为发现她具有一种内在而强大的“智能”，不要被她平时简单质朴的外形所迷惑，一旦其发挥出自己的能力，可以产生巨大的影响力或者破坏力，当然这种破坏力往往是自损。总之，在她温婉圆滑、如诗般清纯的外表下，是一颗闷骚率性且豪放不羁的心。

狮子女——JAVA 语言

很多人都在使用 JAVA，自从诞生以来就以绝对的霸气让全球的互联网都天天侧漏。而 JAVA 更是曾经长期霸占编程语言流行榜首位。JAVA 的卖点之一是平台无关性，但不要忘了为了运行 JAVA 的代码需要一个非常耗费系统资源的虚拟机！最生气的是，必须强制安装平台相关的“虚拟机”才可运行。那么，这样霸气甚至是不讲道理的“语言”何以如此流行？因为她让那些自视甚高的程序猿找到了一种发泄的渠道，认为一旦征服了 JAVA 就可以天下无敌了。却没想到其实内心依然是个自卑的人。

处女女——Pascal 语言

很多 70 后的程序猿是学 Pascal 入门的，这种架构化严谨而独特的语言，让很多人学会了严谨的科学态度。但是不要忘了，这样严谨甚至苛刻的语言，吓跑了很多向往自由和风格不羁的程序猿。更重要的是她独特挑剔的品味，敏感细腻的风格，以及纠结 嗦的“编译环境”都让很多人望而却步，且 Verbose 模式常常打击人的成就感。在“编译”二进制和“底层”需求方面，她保守而不讲人情，甚至条件苛刻，往往一次莫名其妙的 Error 仅仅只是因为你糟糕的 Coding Style。

天枰女——Ruby 语言

Ruby 是红宝石的意思，所以这样的女人也如此，看起来像红宝石一样高贵典雅，其实外强中干。不仅运行性能较差，而且应用领域多限于互联网应用，更无奈的是她完全“面向对象”，对过程一概忽略。和这样的“语言”相处，最重要的也许是给她以虚幻的红宝石幻象，让她时刻保有高贵典雅的气质，避免暴露出外强中干的实质，更要防止暴露出她为了提高性能，而提出的过高“底层”需求。

天蝎女——C 语言

这是一个令人着迷的“语言”，也是一个难度很大的“语言”，即使是一个 Hello World 也是一系列库、源程序经过编译、链接以后的结果。最关键的是，她的魅力就在于其可以通过“指针”直接满足“底层”需求，当她有高涨■

本文更详细内容，请查看：

http://developer.51cto.com/art/201211/363030.htm

学会像一个函数式程序员那样思考

函数式编程已经开始流行起来了，它宣称具有更少的缺陷和更高的生产力。尽管很多开发者曾经尝试过函数式编程，但是依然无法理解是什么让函数式语言如此引人注目。通常学习一种新语言的语法是相对容易的，但是学会以不同的方式去思考却很难。Neal Ford 在他的函数式思考专栏里面介绍了一些函数式编程的概念，并讨论如何在 Java 和 Groovy 中去运用这些概念。新视野 “软件架构” 定义的决策因素

在开始进入正题之前，我们先来做一个比喻。假设你是一个伐木工人，你拥有一把这个森林里最好的斧子，而它使也你成为了当地最有生产力的伐木工人。 某一天，有人向你展示并称赞了一个新的伐木工具 -- 电锯。由于销售人员是一个非常能推销的人，所以你买了一把电锯回来，尽管你并不知道如何去用。于是你尝试像以前砍树那样的来回摆动去锯树。并且你很快得出了一个结论这个新式的电锯毫无用处，于是你又重新拿起斧子去伐木。一直到有人过来并给你演示了如何去运转电锯，你才明白这里的不同。

你可能联想到了用函数式编程来代替故事中的电锯。但是问题在于函数式编程是一种全新的编程模式，而不是一门新的语言，语法只是一个细节问题。而最不同的地方是要如何以不同的方式去思考。而我作为一名 “电锯演示者” 和一个函数式程序员来到了这里。

欢迎来到函数式思维专栏。这个系列将探索函数式编程的话题，但是并不仅仅局限在函数式编程语言有关的内容上。正如我描绘的那样，以函数式的方法来写代码涉及到了设计，权衡，代码重用和其他一系列的观点。我会尝试着以 Java（或是类 Java 语言）的方式尽可能多的展示函数式编程的概念， 进而演示一些其他语言的能力 – 那些 Java 不具有的能力。当然我不会直接切入的非常深，然后讨论一些时髦的事物。取而代之的是，我会逐渐演示一种新的思考问题的方式（或许你已经在某些地方用了，但还没有意识到）。

在接下来的两部分里，你可以把它当作是有关于函数式编程话题的一个旅行。其中的某些概念将会有大量的细节，在这个系列中我会用更多的情景和细节去描述。在旅程开始前，我将带你看一下一个相同问题的两个不同实现，一个用传统的方式来写，另一个使用更多的函数式方式。

数字归类

谈论两种不同的编程模式，你必须用代码来做比较。第一个例子是我另一本书《The Productive Programmer》和《测试驱动设计 1，2》两篇文章中的一个变体。我选取了少量的代码，因为在这两篇文章里已经深入的分析了这段代码。这些文章对这个设计所做的称赞并没有错，但我想在这里进一步提供一个不同的设计意图。

问题的需求是这样的：假设给定任意一个正整数都大于 1，你必须按照完美的，过剩的和不足的进行归类。一个完美数正好是它所有整除因子的总和。同样地，一个过剩数的所有整除因子总和大于该数，而一个不足数的所有整除因子总和

小于该数。

快速数字归类器

列表 1 中的类(NumberClassifier) 满足所有这些需求：

这段代码有几处地方需要关注一下：

它拥有大范围的测试(有一部分我是为了讨论测试驱动开发而写的) 注：这条所说的测试位于作者另一篇文章中。

这个类由大量的紧耦合方法组成，在它的构造函数中拥有测试驱动开发的边际效应。

在 calculateFactors() 方法里内嵌了性能优化算法。这个类的主体是由采集因子组成，因此我可以在之后对它们进行求和并进行最终的归类。整除因子总是以成对的形式被获取。例如，如果这个数是 16，当我采集的因子为 2 时，我就能得到另一个因子为 8，因为 8x2=16。如果我获得的因子是成对的，那么我只需要去检查那些有平方根的数，这就是 calculateFactors() 方法所做的事情。

更多的功能归类

使用相同的测试开发技术，我创建了一个修改后的版本。列表 2，更丰富的功能数字归类器

这两个版本的类尽管差别细微但是很重要。最主要的区别是例 2 的版本缺少了状态共享。消除状态共享在函数式编程中是比较受欢迎的一种抽象手法。作为跨方法共享状态的替代方案，我采用直接调用的方式来消除状态共享。从设计的角度来说，它让 factors() 方法变的更长，但是它也防止了 factors 字段暴漏到方法之外。注意，例 2 是完全由静态方法组成的。在方法间不存在知识共享的问题，因此我可以在更少函数范围上做封装。一旦你给它们输入参数和期待值，这些方法都会工作的很好(这个是一个纯函数例子，这个概念我在将来会进一步探索它)。

函数

函数式编程属于一个宽泛的计算机科学范畴，它已经受到了极大的关注。有新的基于 JVM 上开发的函数式语言(如 scala 和 clojure) 和框架 (如 Functional Java 和 Akka)，它们都声称能够带来更少的缺陷，更高的生产力，更易读，更赚钱等等。相比驻足门外的去解决函数式编程这一大话题，我更愿意将注意力放在一些概念以及这些概念衍生出来的话题上。

函数式编程的核心就是函数，正如在面向对象语言里面类是主要的抽象那样。函数形成了处理过程的基础，同时它具有其他传统语言没有的一系列特性。

高阶函数

高阶函数可以将其他函数作为参数或者作为返回结果。这在 Java 语言中是无法想像的。最接近的方案是你使用一个类(通常是匿名类) 作为执行方法的“持有者”。Java 没有独立的函数 (或方法)，因此它们不能作为返回值或参数出现。

这个能力对函数式语言来说很重要的原因有两点：第一，拥有高阶函数就意为着你可以在如何连结语言元素上作出一个假设。例如，你可以构建一个机制来消除一个类继承体系上的一大堆方法，通过遍历列表并对每一个元素应用一个 (或多个) 高阶函数来实现。(我将展示一个简短的例子给你) 第二，通过将函数作为返回值，你有机会去创建一个高动态，适应性的系统。■

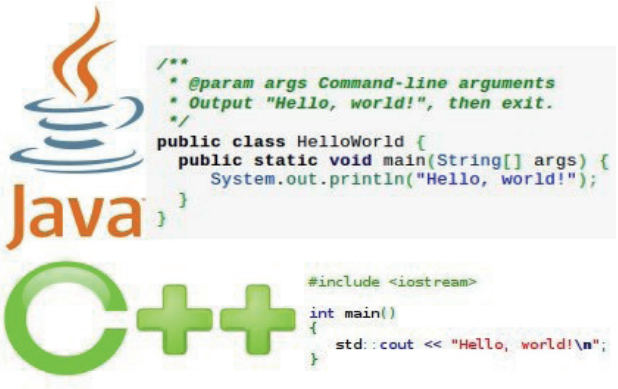
本文未完，剩余部分及代码请参考：

http://developer.51cto.com/art/201210/361398.htm

C++程序员和Java程序员的差异

首先说明，这几年接触的最多的程序员就是 C++ 程序员和 Java 程序员，这只是我自己的一点体会而已，一个人的眼光难免是浅薄的。不过，欢迎拍砖，也欢迎对号入座。

首先说明，这几年接触的最多的程序员就是 C++ 程序员和 Java 程序员，这只是我自己的一点体会而已，一个人的眼光难免是浅薄的。不过，欢迎拍砖，也欢迎对号入座。



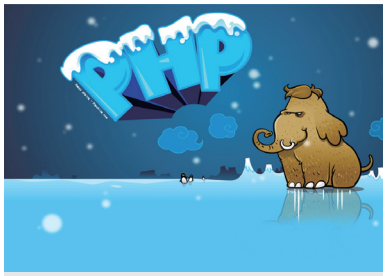
从对基础知识掌握的程度来说，例如数据结构和算法，C++ 程序员要比 Java 程序员牢靠得多。究其原因，大概是因为 C++ 程序员需要经常自己实现那些项目基础设施，但是 Java 程序员大多拿来即用。公平地说，C++ 不是一门适合程序员初学者的语言，但是 Java 是。C++ 程序看起来确实要比 Java 程序稍难懂一些，C++ 程序员往往思维更加严密。另一方面，我也发现内地的程序员要比沿海和发达城市的程序员基础扎实，而且低调内敛，不那么浮华，但是就是善于解决那些难啃的问题。

从视野的角度来说，Java 程序员往往要更宽泛一些。由于语言本身上说，Java 将很多 C++ 逃不掉的陷阱和坑洞都填补了，仅从语言上说，无论是入门还是使用都显得更加简单，程序员就有更多精力来关注语言本身之上的事情。而且，Java 领域的类库明显要丰富得多，所以在考虑某一个设计实现的时候，Java 程序员很喜欢到处寻找可以拿来即用的类库，但是 C++ 程序员习惯于自己去做其中大部分的事情。对于面向对象设计，模块和组件的设计，Java 程序员往往要掌握得稍好一些。

在读书的时候，很多程序员都会给自己订立一个方向，有的喜欢写底层实现，有的喜欢研究人工智能，有的喜欢做企业应用，有的喜欢涉足互联网……就像对 C++ 和 Java 等语言的选择一样。当然，工作有时候并不如心中所愿，这方面可能和自己最初的方向并不一致——有人选择继续，发现这个活儿也不错；有人就会选择放弃，继续执着地寻找自己喜欢的事情。我有三个当初和我一起学 Java 学 J2EE 的朋友，现在一个成了 DBA，一个卖水果去了，还有一个在创业搞机顶盒。

当然，也有很多程序员天天都在纠结：学什么更赚钱？ C++ 还是 Java 有前途？前端还是后端有前途？转管理还是转咨询？也有人这样问我，对于这样深不可测的问题，为了避免显得我没文化、见识浅，我总是一笑置之。

■



PHP 5.4.8和5.3.18发布

PHP 开发团队今天发布两个更新版本，分别是 5.4.8 和 5.3.18。这两个版本修复了超过 20 个 bug。

所有的 PHP 用户建议立即升级到 5.4.8，或者至少也要升级到 5.3.18。

两个关键的问题修复：

Fixed bug #63111 (is_callable() lies for abstract static method)

Fixed bug #61442 (exception threw in __autoload can not be catched

源码下载：http://www.php.net/downloads.php

Windows 二进制版下载 http://windows.php.net/download/

完整 bug 修复记录 http://www.php.net/ChangeLog-5.php

■ PHP 是英文超文本预处理语言 Hypertext Preprocessor 的缩写。PHP 是一种 HTML 内嵌式语言件，在服务器端执行的嵌入 HTML 文档的脚本语言，语言的风格有类似于 C 语言。

谢恩伟:中国MVP作用大

在微软 MVP 十大城市社区巡讲活动上，51CTO 记者有幸采访到微软大中华区副总裁兼市场战略部总经理、微软大中华区首席云计算战略官谢恩伟先生。

今天这个活动就是微软 MVP 们自发组织的活动。在这些活动中，通过 MVP 的技术分享可以让更多的技术人员了解到微软的新技术、新动向。

谢恩伟先生在专访中表示，微软希望更多来自草根 IT 人的声音能出现。也希望在中国的 250 多位 MVP，能更多的参与微软新技术、新产品的研发过程。同时微软期待更多的技术人员能加入到 MVP 的队伍中来。

http://developer.51cto.com/art/201210/362633.htm ■

■ 微软最有价值专家 Microsoft MVP（Most Valuable Professional），是美国电脑公司微软的一个年度奖项，专门授予那些在微软相关技术社区中贡献突出的专家。

架构师眼中的MySQL开发模式

2007 年夏天，当我第一次以资深 MySQL 架构师的身份——这份工作一直伴随我走到现在——参与到企业业务当中时，议程备忘中的第一项内容就是讨论开发模式问题。2005 年，也就是当时的两年之前我们推出了 5.0 版本，而接下来将要接棒的新一代 MySQL 5.1 还要一年多才会面世。但大家一致对 MySQL 5.0 与 MySQL 5.1 的开发模式感到不够理想，因此我们就如何做出改善问题展开了大量讨论。

五年时光匆匆而过，时至今日我们使用的是被称为“里程碑”（Milestone）的发布模式，并已经开始以此为基础努力打造第二个 GA 版本。

新旧两套开发模式间的不同之处

现在我们来聊聊新旧两种开发模式之间的几项主要差异。

我认为目前我们所使用的开发模式，其最重要的特性就是整个开发流程以 2 到 4 个月为周期始终处于随时可发布状态，并且能够稳定地按照每 6 个月推出新 GA 版本的进度进行运作。对于可发布状态，我们的定义为：产品质量严格符合候选发布版的相关标准。在过去的开发模式中，我们的标准相对比较宽松，所谓可发布状态是指当前版本到达候选发布要求——但由于在测试阶段仍然可能有新功能不断被添加进来，因此整个发布流程周期常常长达两年，而不是现在的 2 到 4 个月。

新模式给 MySQL 新代码带来的影响

那么这种新模式会给我们的 MySQL 新代码开发工作带来哪些影响呢？在我看来，这意味着新代码必须以小规模模块的形式添加到产品中来，换句话说新代码中不能混杂过多旧代码，因为这可能会带来新的未知冲突及不稳定因素。总之，如果需要向 MySQL Server 中某个复杂程度较高的区域添加新功能，我们会首先对该区域进行重新设计，然后才着手实施功能添加工作。如此一来，我们就在某种程度上保持了开发流程的稳定性始终拥有最高优先级。而在过去，我们向 MySQL Server 中添加新功能时往往会造成系统稳定性优先级别的下降，这对于一款开发模式而言当然不是什么好事。

有趣的是，随着稳定性在开发流程中的重要性得到进一步增强，我们反而获得了放手处理新功能的广阔空间！不过有时候我们未必能按照自己的预期快速开发出特定的新功能。如果一项新功能会对 MySQL Server 中复杂性较高的某些代码造成干扰，那么为了系统稳定性的考量，我们会事先进行一系列准备工作及测试流程，这是此类新功能出台的必要前提。但是对于大部分与 MySQL Server 复杂区域不相关的边缘新功能来说，整个添加过程则更加顺畅同时轻松愉快。

新模式下的主要功能开发流程

接下来要谈的是如何在这套模式中处理主要功能的添加工作，其中最需要注意的因素在于如何保证新代码拥有良好的排布结构。也就是说在添加新功能时，我们不能简单地从编写代码着手，而应该首先思考现有代码结构能否与正在开始的新功能紧密契合。举例来说，我们在 MySQL 5.6 版本中添加了众多新功能，而 LOCK_open 正是其中之一。

架构师眼中的 MySQL 开发模式 II

如果按照以往传统的 MySQL Server 开发模式，我们会直接将新功能添加进来，然后检测它会与哪些现有 MySQL Server 功能组件发生冲突，并确保所有已经确定的新问题在新的 GA 版本发布之前得到解决。但现在我们的思路更先进、处理方式也完全不同。首先我们会建立一系列重新设计项目，将 LOCK_open 主功能完全隔离起来，这样就保护了运行中的元数据免遭破坏、同时也为连接中的元数据提供了缓冲措施。在以上步骤的执行过程中，我们可以继续进行手头的其它 MySQL 5.6 开发工作，而且原本非常重要的 LOCK_open 功能更新能够以小补丁的形式添加进主系统当中，同时也让功能升级变得更加安全。

新开发模式的灵感来源

我们之所以选择这种全新的开发模式，其灵感显然源自以 Linux 内核为代表的其它几个典型开源项目。它们的成功为我们的新思路提供了指导与借鉴。

从旧模式向新模式的转换

尽管新模式的优势如此明显，但从旧模式转换为新模式的道路却并不平坦。在决定发起变革之后，我们制定了新的开发流程，并在其中添加了多项意义重大的新功能。然而流程的出台并不意味着实际工作的顺利进行，概念与可发布状态之间仍然相隔十万八千里，因此我们做出了“一个艰难的决定”——以 MySQL 5.1 开发流程为基础将一切过去推倒重来。这套新流程是我们首次在开发过程中使用“里程碑”模式的尝试，但原先的模式中同样包含很多意义重大的理想方案，因此我们需要一步步将遗留资源与新规划加以整合。整个过程持续了数年之久，由于新的开发流程对于新功能的运作效果及稳定性要求极高，因此进度不快也在情理之中。不过现在我们终于大功告成，所有有价值的功能都已经被迁移到新模式当中，其它的则要么直接弃用、要么利用其它方式加以替代。

对于未来 MySQL 开发的影响

那么新模式的引入到底会在当下与未来给 MySQL 开发工作带来哪些影响呢？首先要强调的是，我们在 MySQL 5.6 中添加了 200 项新功能，这表明新的开发模式确实能为我们的用户及消费者带来大量激动人心的革命性升级与改进。希望这套开发模式能够继续指引开发工作顺利开展，我们也会继续努力为 MySQL 技术社区带来更多更具吸引力的功能。作为一位架构师，我非常自豪地关注着 MySQL 技术团队的一举一动，也为他们针对 MySQL 架构做出的改善而深感骄傲。相信随着新模式的逐渐完善与新功能的持续加入，MySQL 会为全世界使用者带来更加光彩夺目的特性。■ （译者：核子可乐）

Visual Studio 2010 提高效率小技巧

当你在光标停留行使用快捷键 Ctrl+C，X，L 时，可以复制，剪切，删除整行内容。当然，右键也是可以的。

跟平时的复制，剪切，删除就是选中和没选中代码的区别而已。

如果你想交换上下两行，你可以使用快捷键 (Shift+Alt+T)，前提是光标要停留在上面那一行。替换之后，光标会一直跟随原本的那一行。

“菜鸟”程序员与Ruby第一次接触

Ruby 到底火不火，目前难以下结论，不过从招聘方面来看，使用的公司还是少数，从薪水来看，相对于其他，还是蛮高的，可能主要原因是因为学的人少吧。编程语言都有通性，多学一门语言也不是坏事，在朋友推荐下，也粗略看了一下 Ruby 的基本语法，个人由于有 C,PHP,js 方面基础，所以看起来也不费力。

Ruby 给我的第一印象是语言精练，代码精简，不需要写分号，也不需要像 Python 那样严格遵循格式，这是其他语言不一样的。

至于语法都是大同小异，最后接触了一下 Ruby on Rails 框架，学习 Ruby 主要就是学习此框架，绝大多数 php 框架都有多多少少山寨 ROR 的结构。总的来说，框架提供了大部分功能，我们只需要自己去继承即可，开发人员主要把精力放在设计上和业务处理上，时间短，个人还没有看出 Ruby 开发的优势，接下来叙述一下整个安装 Ruby 和 Ruby on rails(简称 ROR) 安装以及开发实例 .

我也是菜鸟一枚，所以对 Ruby 叙述有误请看客包含。开发第一步就是配置环境，我主要是在 windows 下学习，最优的开发环境是 Mac ox 和 Linux, 这方面配置比起 windows 更简单，只需要使用命令即可完成，有兴趣的可以搜索 . 下载 Ruby for windows 版本 (http://Rubyinstaller.org/), 接着安装，默认就安装到 c 盘 Ruby 下面吧。接着在开始菜单找到 Start Command 的命令窗口，这个是模拟 linux 环境的命令窗口，首先我们看到的是 Ruby 版本，查看的命令是 Ruby –v, 我们再来查看 gem –v 版本如果不是 1.8 以上，那么就需要更新，命令是 gem update ––system, 接着安装 Ruby on rails ,gem 其实是一个 Ruby 写的应用程序管理软件 ,rails 也是一个 gem 包，接着要等待很长一段时间来安装 rails, 安装完成以后，我们输入 rails –v 来查看版本 现在是 3.2.4 最新版本，现在基本就搞定了框架，我们可以来测试创建一个 ROR 应用程序，创建命令 :rails new C:\weblog, 这里 new 后面是路径 ,weblog 是应用程序名称，建议使用英文 . 等待一下，一个应用框架就建立好了，我们打开文件夹发现里面有很多不同名字的，从意思上基本知道都是做什么的，以后的开发主要在 app 这个文件夹，打开以后发现里面有 MVC 模式组成的文件夹，这个就是以后学习的重点，到这里为止，前期准备工作已经做好了，接着我们就来演示服务器运行 .

用 PHP 我们知道服务器可以是 apache,nginx,iis 等，当然 Ruby 也是支持的，不过我们在此处使用 Ruby 自带的小型服务器，输入 rails server 就可以启动 WEBrick 服务器，如果启动不了，使用 gem install WEBrick 来安装，其他遇到没有安装的 gem 包时，也可以采用此命令来安装相应的 . 命令简写 rails s 也是可以的，等几秒就启动啦，这时候浏览器输入 http://localhost:3000/ 就可以启动应用程序了，是一个 welcome 页面他的文件实际位置在应用程序目录的 public/index.html 里面我们要改成自己的启动

“菜鸟”程序员与 Ruby 的第一次接触 II

目录呢，就需要重命名 index.html 为 index2.html, 接着创建一个控制器，需要另外再打开一个命令窗口，输入 rails generate controller welcome index, 意思是创建一个 welcome 控制器 ,index 是默认启动方法 (动作),app/views/home/index.html.erb 就是模版文件，在里面输入 hello, 他已经自动创建好了，我们再 config/routes.rb 文件，更改里面默认启动方法，找到 root :to => 'welcome#index' 类似语句，去掉 # 注释，我们现在刷新一下发现 hello 显示了，但是标题和 head 部分完全不是模版里面的，其实这部分在 app\views\layouts\application.html.erb 文件里面，打开就可以看见了，我们可以更改这部分公共文件 . 再次刷新一个你自己创建的控制器和模版就显示了，到此，一个自定义控制器完成了 .

如果你想快速写一个博客，使用 rails generate scaffold Post name:string title:string content:text 命令就可以生成了，不过还是建议自己来定义，具体的项目开发，目前视频资料比较少，也都是在网上找一些教程自学 ,Ruby 是完成面向对象的，比如数字，字符串布尔值都是对象，学习基础语法，你可以在命令窗口输入 irb 就进入到了 Ruby 环境 ,exit 退出来，如果你想写在文件里，比如在 c 盘建立 test.rb 文件，第一行输入 #encoding:utf-8, 文件也保存成 utf-8 编码，这样可以支持中文，比如输入 arr =[1,3,4] 定义数组 puts arr 输出数组保存，然后使用 cd 或 cd.. 切换到文件目录 c, 输入 Ruby test.rb 就可以运行刚才的命令 . 好吧到此基本讲述完成在 windows 下学习使用 Ruby 和 Ruby on rails, 文本讲述的不使用任何集成环境，网上有几个集成环境，但是软件版本很老，更新以后才能使用好像都是 rails2 版本，和 rails3 区别过大，这样按步骤来安装也不是多难，就这几个命令，照猫画虎呗。

Ruby 方面还比较少，可能和使用者过少有关，有这方面学习的朋友欢迎交流，粉我任何联系方式都可以 .http://guides.Ruby-china.org/getting_started.html 这是一个非常好的例子用很简短代码完成的项目例子 .

最后提供几个文中命令

irb 进入 Ruby 命令编程

Ruby 命令

Ruby -v 版本

gem -v

gem update --system 升级

gem install rails 安装 ror

rails new c:\web 创建应用

rails server 启动 WEBrick 服务器

rails generate controller welcome index 创建 webcome 控制器■

程序发布前应该发现的错误

较大的响应输出，应该是容易理解的，那就是：服务端返回的结果太大了。我们可以想像一下较大的响应输出意味着什么。

浏览器显示一个很大网页，是不是比较慢？是不是会花费较长网络传输时间？是不是也要较长的生成时间？如果这个很大网页的结果来自于数据库的查询结果，会不会给数据库也带来较大的压力？产生这种情况可能由于一条 SQL 查询引起的：select * from XXX where name=@name

编者按

详细设计是V模型或者瀑布开发中的一个重要的环节。这个阶段负责把概要设计进行细化，并为代码书写作出指导。可以说是一个承上启下的重要环节。

详细设计成软件开发过程中的浪费

但是现实的情况真的如此吗？我们来反思一下：

(1) 详细设计和代码的吻合程度有多高？

假设在项目中，代码在测试后修改完毕提交后，并不修改详细设计，则详细设计和代码之间并不吻合，并且很大程度上，吻合度会非常低。

如果详细设计和最终的代码并不吻合，那么这样的详细设计并不给将来维护带来任何帮助。

如果详细设计并不能给后续带来帮助，为什么要书写它呢？

因为——详细设计是用来指导代码书写的。

(2) 详细设计对代码的指导意义有多大？

详细设计的类图是用来定义类框架之间的关系的；其中的时序图（有时也用流程图）是用来定义方法之间的调用关系的。

如果说详细设计是这么定义的，那么为什么不直接用IDE写成代码形式？

因为——详细设计的过程是需要记录文档以备后查的。

(3) 详细设计是怎么复查和修改的？

详细设计的复查是通过对书写完成的详细设计文档进行阅读和审查，并指出其中可能出现的错误和遗漏。

然后针对提出的问题进行修改，直到修改完成为止。

为什么要这样复查和修改呢？因为详细设计的质量提高有助于早期发现问题。

(4) 既然详细设计是用来指导代码书写的，为什么还需要后续的测试？

换句话说，为什么详细设计不能100%正确。

这么问并不是要求把详细设计100%书写正确——因为这是个不可能的任务。

详细设计是一个猜想的过程，其复查和修改也都是在猜测中完成的。

详细设计到底做成什么样才能够更有效地指导代码书写呢？

(5) 详细设计的完成是怎么定义的？

详细设计的完成指标是：详细设计的页数达到若干页，每页复查发现的问题达到多少个。

详细设计是用来指导代码书写的。为什么不从指导代码书写的方面进行指标定义？

详细设计还有问题残留的时候怎么就开始代码书写了？

现实情况是：详细设计的完成是以项目经理的喜好决定的——往往是时间压力决定的，还有时间就继续写；没有时间就算完成了。

(6) 为什么觉得详细设计是必要的过程？

因为是规定的。因为别人都这么做。这应该不是答案吧？

那么到底应该怎么书写详细设计呢？答案是：不写！理由如下：

(1) 详细设计的职责不明确

详细设计成软件开发过程中的浪费 II

详细设计名将概要设计细化，并指导代码书写。但是反观其阶段结束时间不明确，并且阶段结束的判定标准也没有对如何指导代码书写进行定义，很难说详细设计真的是用来指导代码书写还是将概要设计细化的。

(2) 详细设计没有生产有价值的产物。

据统计，详细设计在项目开发过程中所消耗的工时基本上占编码和单元测试的一半。但是它的产物——UML 图可以通过直接书写代码，然后从 IDE 导出生成，这个过程只需要几秒钟。

但是详细设计还是要做的。这不是和前面矛盾吗？不！

上面说的是写，这里说的是做。

有些处理的条件很多，不是三言两语能够说清楚的，这时候就需要详细设计。

比如：一个由两组条件决定的处理。从需求角度山来说，通过画成二维表可以描述清楚其各个条件组合下的行为

	a	b	c
1	a1	b1	c1
2	a2	b2	c2
3	a3	b3	c3

针对这种情况，如果写流程图，那么代码也将会按照流程图那么书写，代码会变得很冗长。所以流程图并不适合做详细设计。这种情况可以通过类的书写来完成。

假设行条件是时间范围 (TimeScope)，分别代表过去 (Past)，现在 (Present) 和将来 (Future) 而列条件是状态 (State)，分别代表申请 (Apply)，批准 (Approval) 和拒绝 (Decline)

在类的设计时，以时间范围为主轴，状态为辅轴，那么类定义为：

```
interface TimeScope {
    public void apply();
    public void approval();
    public voi decline();
}
```

则 Past，Present 和 Future 分别实现相应的方法就可以实现上面需求定义的矩阵。

这种方法叫做桥接模式 (Bridge Pattern)

而这个过程并不需要留下文档。

同样大多数情况来说，根据需求可以直接生成代码。而详细设计是一个可以简化到只要几分钟就可以完成的过程。并且，从质量的角度来说，并没有损失。

设想一下，如果一个项目可以略掉详细设计过程，其可以带来的节省有多大。

如果客户非要要求详细设计怎么办？

雇佣比软件工程师便宜的文档人员来根据代码反写详细设计——因为类图都可以通过工具生成——所以，所需要的文档人员也很少，工时也很少。■

CSS 小提示

不需要给背景图片路径加引号，比如：

background:url("images/***.gif") #333;

你应该把引号去掉

background:url(images/***.gif) #333;

如果你加了引号，反而会引起一些浏览器的错误。

物理学家眼中的世界：编程的未来

作者 / 四火

最近在看一本书，加来道雄（Michio Kaku）的《物理学的未来》，第一、第二章是程序员更加关心的，对于下一个 100 年计算机和人工智能未来的预测。想想计算机发展短暂的历史，这些发生了的翻天覆地的变化，似乎都在弹指一挥间。谁的大胆预测可以那么准确？无论如何，书中对其这样几个猜想令我记忆深刻：

- 因特网眼镜和隐形镜片
- 无人驾驶汽车
- 摩尔定律结束
- 通用翻译器
- 全息摄影和三维影像
- 意识识别
- 有意识情感的机器人
- 模拟大脑

这是物理学家眼中的世界（另外推荐他的另一本书《平行宇宙》），激动人心；另一方面，我回想起小时候无比痴迷的机器猫，小小四维空间袋，寄托了孩子多少纯真的梦想，有多少神奇的道具已经成为现实……

但是程序员要说的看法，尤其在自己熟知的领域，我们不谈语言的发展和趋势，这些留给专家去做吧——不妨把目光放长远一点，100 年后的程序员，他们都在做怎样的事情？ 100 年后的编程，会是怎样的一种劳动？

人人都会编程

微博上，有朋友对于 HTML5 实现的 web 操作系统评论道“断网就是废物一个”，但是他并没有意识到，很快网络就将如同现在的水、电这样一样，是人正常生活不可缺少的基础设施。

类似的，编程，也将是未来人们日常生活的必备技能，如同写字、阅读一样。编程并不非得指写那些非程序员看不懂的奇形怪状的代码。你把衣服放到洗衣机里，设定好水量中等，浸泡 20 分钟，洗涤 20 分钟，漂洗 3 次共 15 分钟，再甩干 3 分钟——这，就是编程，你做的仅仅是按几个按键，把这几项工作组合起来。

再如 ifttt 这样的网站，你都可以实现编程的分支功能了——如果明天天晴的话，就发给你一条短信，去爬山。完成这样的功能，你根本不需要是程序员，你只要会操作电脑，会上网就可以了。

互联网的资源，将被得到更好地组织和获取，以 YQL（Yahoo! Query Language）为例，你可以体会到这一点：

```
select * from html
where url='http://www.dangdang.com/'
and xpath='//ul[@id="homepage_promotion
_count_ul"]/li/p[@class="name"]/a'
```

它做了这样一件事：从当当网的页面去获取数据，而数据的路径通过 XPath 表达式给出。如此一来，你可以感受到，整个互联网就变成了一个超级大型的数据库。当然，这样的语法还是不够简单，希望能看到类似 ifttt 的应用出现，目的却是让不会编程的人也可以轻松从互联网这个大型数据库中查询自己需要的东西。

另外，未来需要普通人掌握的编程技能也不

物理学家眼中的世界：编程的未来 II

尽相同，就如同现在年轻人和老人的阅读技能大不相同一样。但是可以确定的是，生活中会充满编程的行为，让机器替代自己做更多的事。

所见即所得

好吧，在这里我谈这个话题也和我的启蒙编程语言是 VB 有关。你也许和我一样，谈到所见即所得的时候，想到很多编程语言、IDE，甚至包括 FCKeditor 这样的富文本编辑组件。Google 已经做了这样的尝试，App Inventor 就是这样的东西，它是为手机端准备的编程软件，你可以看看这样的宣传视频：

略（请进入文章页观看，低下有链接，谢谢。）

上面这则视频似乎只是针对非专业程序员的傻瓜式工具，那么再来看看这个在网上已经广为流传的 Bret Victor 的神一般的演讲，题为《Inventing on Principle》，第一次看的时候，你一定会像我一样惊讶地合不拢嘴：

略（请进入文章页观看，低下有链接，谢谢。）

所见即所得使得编程的过程更贴近人最自然的思维，而一张丰富画面所传递的内容远远大过枯燥的代码行语义和数值。

编程范型的进化

相较于硬件的摩尔定律，软件的发展似乎真的是“太慢了”，相较于硬件淘汰的速率，几十年历史的编程语言却可以长盛不衰地存活下去。好在软件的发展也是有驱动力的，软件的复杂性就是直接驱动力之一。想想现在做一个普通网站的代价，和十五年前比较，我们能省做多少功。

很多时候程序员会觉得，算法还是不容易转变成代码，即便是简单的算法，思路简单的纸上实现，变成代码却比较冗长。我觉得大部分情况下这不是你编码技巧的问题，而是编程语言的问题——换句话说，如果你使用一种合适范型的编程语言，兴许就可以轻松解决这个问题——即便这样的语言并不一定好找，并不一定容易设计。

我们都知道从过程式编程到面向对象编程的进化，可是如今常用的编程范型已经远远超出这两者了，例如声明式编程、面向方面编程、基于规则的编程等等，我们的固有思维模式一次有一次遭到挑战。

以 Prolog 语言为例，它是由事实和规则组成的，我们先告知程序这些已知的事实和规则，再去询问程序一个需要推断的问题，让它给出推断的结果。比如：

```
love(you, dog).
love(he, dog).
love(she, cat).
friend(PA, PB) :- \+(PA=PB),
love(PA,Animal), love(PB,Animal).
```

我来解释一下：

给定了三个事实：你爱狗，他爱狗，她爱猫；

给定一条规则：对于人物 A（PA）和人物 B（PB），如果人物 A 和人物 B 不是同一个人（“\+”表示取反），人物 A 爱动物 Animal，并且人物 B 也爱同一种动物 Animal，那么人物 A 和人物 B 就是朋友（friend）。

好，现在来询问程序一个问题■

本文更详细内容，请查看：

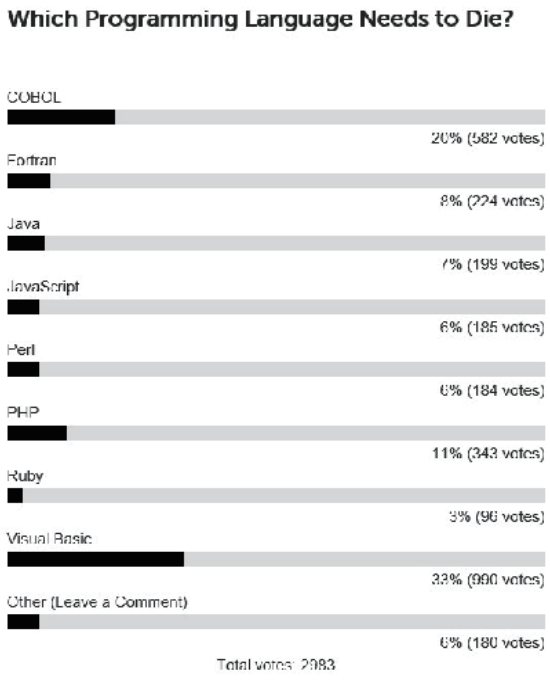
http://developer.51cto.com/art/201210/360936.htm

动态编程语言遍地开花：浅析Ruby

作者 / 小林

【51CTO 独家特稿】随着计算机的快速发展，编程语言也越来越多，在 10 年因为开发速度的问题，Java 成了编程语言的老大，随着 Java 不断的臃肿与复杂，开发者也不断的去寻找，尝试各种新的语言，也因近年动态语言有后来居上的趋势，Ruby 作为具有动态类型的解释型面向对象语言因此而被开发者热捧。革命的年代已经结束。Ruby 从 1995 年至今已经成熟成为了编程界的主流语言，至少主流在向 Ruby 前进。

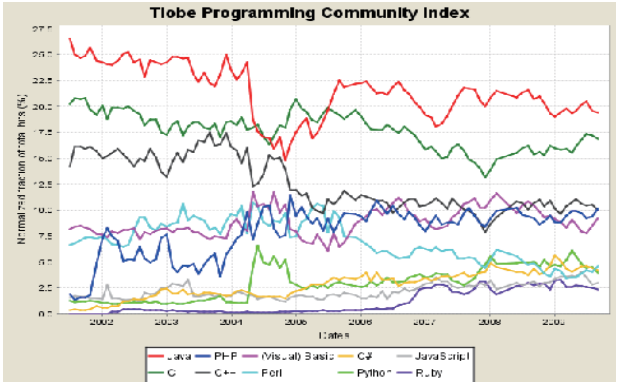
当然，在这个充满偏见的语言战争年代，或许是一场无意义的比拼，这是哪个语言都无法回避的。从复杂性问题本身来考虑，为什么说它受到开发者的热捧？首先我们来张国外某博客发起的一次变成语言的投票。



从 2983 个人的投票中我们看到，仅有 96 人认为 Ruby 会死，也是上面变成语言中最少的一个。在国外，Ruby 已经是主流的编程语言，Ruby 的开发理念认为解决问题的方法可以不止一种，选择哪种取决于程序员的喜好。

在旧金山的独立开发者 Sara Mei 就一直用 Ruby 开发大量的应用，在他看来，Ruby 的魅力在于它是一个使用起来非常舒服的编程语言，它具有许多强大的元编程功能。

为什么选择 Ruby 而不是 PHP 或者 Python 之类的语言呢？



从图片上我们看到了 Ruby 从 06 年开始关注 Ruby 的人渐渐增多，在国外 udemy 博客中 Renee 就做了个详细的对比，重点介绍了三个当今最流行的编程语言。

略（由于图片过长，请进去文章页查看）

从这张图片上看在可用性方面 Ruby 无疑是超越了 PHP 和 Python 这两种语言，PHP 是属于比较传统的，可用性较弱些；Python 相对来说稍微好点；Ruby 的优势在于代码编写优雅、强大和表达性好，可用性极高。■

本文更详细内容，请查看：

http://developer.51cto.com/art/201211/364413.htm

创业者十诫：不选择创新工场理由

无论是创新工场的项目现在遭遇越来越多的质疑，还是它本身现在向纯粹投资机构的转型，都给中国的创投圈和创业者带去了前所未有的冲击和震荡，理想和现实之间总是有着不可逾越的鸿沟，创新工场建立之初的光环正逐渐褪去。

无论是创新工场的项目现在遭遇越来越多的质疑，还是它本身现在向纯粹投资机构的转型，都给中国的创投圈和创业者带去了前所未有的冲击和震荡，理想和现实之间总是有着不可逾越的鸿沟，创新工场建立之初的光环正逐渐褪去。

来自匿名人士 hackerTom 的这篇文章能让我们从一个更加直观的角度向大家讲述一个与以往大家认知截然不同的创新工场和李开复形象，Ta 告诉我们为什么创业者不要选择创新工场。

1. 创新工场不看项目看背景。

作为创业者，你也许是白手起家，并且手上负责着一个不错的项目，并为此辛劳付出，并且年轻有为。但很遗憾，凡能进入创新工场的项目的创始人很多都是以往在 XX 大公司担当过什么重要职位或者 XX 著名大学毕业出生，有着各种“辉煌”背景的人才会有机会被看中，那怕你的项目还没有上线。而对于项目的模式，前景，规模等资料基本上不是关注的重点。

这与创新工场的“帮助中国青年成功创业”和李开复自称的“青年导师”相背而持，基本上里面是没有一个青年创始人或 CEO。其实纵观创新工场的发展，与其说是帮助，倒不如说忽悠某些有才能的青年去加入创新工场，为那些有背景的创始人打工。

所以，如果你没有“背景”，那怕你的项目再好，基本上没有什么机会。

2. 创新工场占据股份远远高于国外的孵化器

根据最近李开复自己的澄清，创新工场平均占据项目的股份只有 17.6%（刚开始接受采访的时候说是 30%-40%）。现在姑且相信平均 17.6% 的股份是真的，但是相对于国外著名的孵化器 YCombinator 占据 2%-10%，500startups 占据 5% 左右等，基本是它们的好几倍，甚至比一般的 VC 占据 15%-25%，天使投资占据 5%-10% 的还要高。作为一个辛辛苦苦创业的你，不管创新工场究竟是孵化器还是投资机构，你会愿意选择把 17.6% 的股份让给如此贪婪的它麽？

3. 创新工场不创新

相信这点基本上不用再这里讨论了，创新工场不创新，不但在国内引起争论，连国外媒体

创业者的十诫：十个不选择创新工场的理由 II

都讽刺，甚至李开复还因此得到了，“李开始复制”的头衔了。

4. 一定要懂得拍李开复的马屁

来源根据：讲述 Google 公司内幕的《In The Plex》http://www.guao.hk/posts/in-the-plex-1.html

– 从山寨城来到谷歌中国总部的 Wesley Chan 感觉谷歌中国办公室非常的“不 Google”，很多人成天就是围绕在李开复身边。有一次开会很多李开复招进来的人在争吵自己的头衔应该是什么，Chan 跟他们说你们就是产品经理。这些人却说在景德镇没人知道这个职位是啥意思，他们更喜欢被称作“李开复特殊助理”之类的，这样别人就知道他们是谷歌中国尊敬领导下的一员了。Chan 对此很愤怒，他说：“这儿又不是白宫！我们的工作是关注我们的用户，而不是李开复！”

– 谷歌中国还有个奇怪的办公室文化，就是谁坐的离李开复近谁就光荣。最后 Chan 实在受不了，走人之。

– 曾经有山寨城来的交换工程师对谷歌中国办公室的评价甚为经典：“帝都和其它世界各地工程中心最大的不同就是，其它办公室一看就知道是 Google，帝都办公室的一看就知道是谷歌中国”。

由此看出，李开复最喜欢就是搞个人崇拜，谁对他最好，谁最会拍他马屁的，谁在李开复眼中是个听教听话的”乖小孩“，谁就能获得李开复的重用。一旦整个公司都弥漫着这种氛围，基本在某个程度上注定了公司的失败。

5. 李开复占据着创新工场举重轻重的地位

现在很多人都知道李开复就是创新工场的 CEO，创新工场最有名的也就是李开复，甚至可以说李开复的名气比创新工场还要大。这是一个非常奇怪的现象，基本上没有多少投资机构或者创业孵化器的创始人的名声比自身创立机构的名声还大的。作为这些投资机构的创始人，在媒体露面的次数不会很多，以保证旗下投资的企业不会受到自己个人行为或者言论所影响。

但是现在，一旦说起创新工场旗下的某个项目，基本上都把它和李开复个人划上等号了，背后的创始人基本上没有多少人能记得是谁。

6. 李开复个人的言论和个人行为太出众

如同上一条所说的，李开复自从从 Google 离职后，频繁出现在国内各种媒体演讲中，并且经常对各种公司发表个人的言论，争辩，甚至和别人打口水仗等等。而且最初从大部分人都崇拜李开复，逐渐变成现在反对其作风的人越来越多，基本上每次 IT 媒体报道和社区下的各种评论大部分都直接针对他做出各种评论和质疑的声音。这不但影响着李开复个人的形象，更是影响着其投资下的各种项目的形象。

作为一个创业公司，你愿意把你的项目加入到如此的言论当中麽？

7. 创业公司的话事权不大

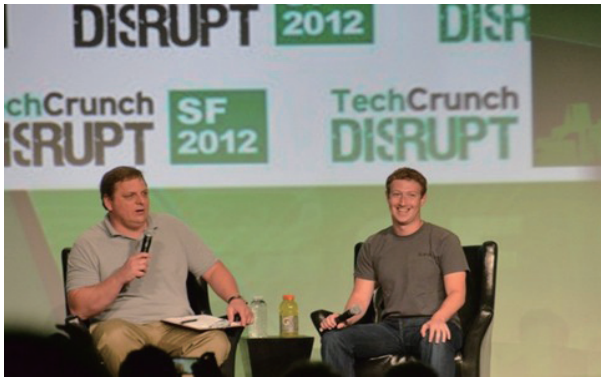
由于创新工场其架构的问题，尽管根据说明只占有 20% 左右的股份，但基本上在里面的创业公司都无形中被创新工场的各种“专业导师”牵着鼻子走，被遭到各种的 “教育”和“指导”，创始人也基本上没有太大的话事权，只好摸摸地接受。一旦创业公司的创始人失去了话语权■

本文更详细内容，请查看：

http://developer.51cto.com/art/201211/363916.htm

HTML 5欧亚大陆冰火两重天

作者 / 楚萧宁



两月余前，关于 FACEBOOK 放弃 HTML5 而改用原生的事被炒的沸沸扬扬，一时间 HTML5 的处境下滑冰点，归其原因是“慢”。Webkit 的解析过程先后需要经过解析、建立 DOM 树、获取对应资源、布局、建立渲染树、绘图到展示。这跟传统的商品流通路径非常相似：工厂、品牌公司、总代理、经销商、卖场、消费者。由于环节太多层层加价，产品到达消费者手里往往价格居高不下。而 HTML5 的代价同样是用户为使用产品而付出的时间成本。原生就好比 F2C，用户直接从“工厂”即本机获取产品 / 服务，原生由于是本机直接处理，要比浏览器一层层处理下来快的多。谷歌数据：网页加载超过 4 秒，25% 人会放弃；手机网页超过 10 秒，50% 用户会放弃，60% 人不再返回；Google 搜索结果慢 0.4 秒，一天搜索量减少 800 万次；40% 移动购物者会放弃加载时间超过 3 秒的网站；亚马逊每天销售额约 6700 万美元，网页延迟 1 秒可导致全年损失 16 亿美元。

一、Fireox OS 问题重重，前景未卜

在 PC 上，网页一样是慢于客户端，但与移动端相比，PC 用户明显是长使用时间，而且有稳定的宽带速度，等待的时间成本相对较小。而移动端是我们众所周知的碎片化消费。让用户在碎片时间里再去消费掉大量的时间去等产品的运行，完全是扯蛋。至此，似乎非 Fireox OS 或 webOS 不能拯救 HTML5 了。但预计 2013 上市的 Fireox OS 是否能如期上市，还是一个很大的问题。就算如预期上市了，Fireox OS 本身还有大量的 BUG，不能支持 android 平台软件的系统也极度的缺乏第三方应用。而 webOS 也早早被 HP 在 2011 年 8 月公开宣布停止开发 (包括 Palm Pre 手机和 Touch Pad 平板)。

二、GREE 为何能够走红 HTML5 之路

有评论说国人把 HTML5 捧的太高，HTML5 的不成熟引发出各种问题，加上 FACEBOOK 的失败，似乎验证了 HTML5 注定没有好下场，可我们似乎忽略了一个在 HTML5 之路上一路走红的“家伙”，那就是日本的社交网站 GREE。非常奇怪的是，我们讨论了几乎一年的 HTML5，却一直很少有人专门提及 GREE。当然，也可能是我见识的少。不管怎么样，这“家伙”一路走

的很 HIGH。2008 年 12 月 17 日，GREE 登陆东京证券交易所创业板，首日开盘价超过发行价 51%，市值达到 1070 亿日元，超过日本第一大社交网站 Mixi 跃居该创业板市场首位。在最近的数据中显示，GREE 的用户已达 1.9 亿，虽然与 FACEBOOK 的 10 亿不能相比。但我们比的不是用户，所以请把目光转回产品本身。GREE 平均 98% 的页面浏览量都来自移动设备，而且也没看见人家的用户不爽网页滚屏效果不流畅。Why? 请注意，我们之前已经解释过了为什么 Web APP 比 Native APP 慢，在 GREE 社交网站上，只有游戏是 HTML5。而在 FACEBOOK 上，HTML5 负责处理了太多的信息加载、图像绘制和数据的加载，这些都不是 HTML5 擅长的。而 GREE 就聪明很多，在大量的数据方面， 他们尽量为平台做瘦身，降低平台本身的停留时间，同时用常规语言编写平台，但在游戏方面，他却采纳的是 HTML5。在 GREE 约 502 款的游戏中，有 30 ~ 50 款是需要下载的，剩下的全部是 HTML5 或 WAP 游戏，而 HTML5 游戏又以卡牌类为主流。混迹游戏的人士都知道日本人爱好卡牌类游戏，体验过的朋友就知道这类游戏刚好画面效果简单，是 HTML5 完全能够支持的。

如此这般，用户在打开 GREE 进行各种社交行为和玩游戏的时候，完全不受影响。而 FACEBOOK 却偏偏露了 HTML5 现在的短板。

三、中国 HTML5 之路该如何走下去

反思中国现在的几只领头羊产品，立刻玩、多泡、e 点游戏，无一不是全部使用 HTML5 开发的，做为中国敢为人先的少数产品，如果因此而夭折，未免令人痛惜。有幸曾与立刻玩平台的制作人一起聊过，在谈到这里的时候，他也表述现在过于追求 html5 在中国不是非常现实的，html5 游戏平台，强调的并非 html5 语言所开发的的游戏平台，而是提供给用户无终端限制无需下载的 html5 游戏。所以平台本身，网站版尽量用常规语言开发，减少平台本身的效果，降低平台本身的停留时间，将用户引导进入游戏产品本身，客户端版本，用原生语言开发，游戏内容采用 html5 语言，下载一个原生客户端，可以体验无限款 html5 游戏。

后记：重内容，轻语言，优先的产品才是王道，HTML5 不能代表一切。一个平台产品本身做的多绚丽不重要，就像是白纸，纸用什么造的不重要，关键是白纸上的图案。

页面构建和JS前端不得不说的那点事儿

作者 / 张传亮

作为微博的页面构建工程师，主要职责就是利用 html&css，高质量的完成静态页面的制作，保证项目的按时完成。而页面需要的 js 效果则交给下游的 js 前端工程师去做。在微博，这两个岗位是分开的。但在大家的思维定势里可能觉得这两个岗位应由一个人来完成最好，毕竟，页面构建工程师写的 html 结构不一定是 js 工程师想要的那种，js 工程师可能有更高效的方式。所以，在页面构建之前最好能与 js 工程师沟通一下，把实现方案确定好。

但在实际项目流程中，当进行到页面构建的时候，产品经理可能还没安排到 js 资源，这时我们只能按照产品的需求和自己的想法去写 html 结构，不仅要考虑到设计稿的还原度、浏览器的兼容性、以后可能要添加的新功能的预见位置，还要考虑怎样写 html 结构能让 js 最省事的完成产品的交互效果。作为一名有责任感的页面构建工程师，表示亚历山大！所以，我们经常查看页面源码时，会发现这样的注释，用来告诉 js 工程师怎么去操作 dom 结构。

```
<div class="content olympic 2012 turn olym on">
<!--不出现turn_olym_off为默认展开  出现为收起-->
```

有时为了做到最简，我们要考虑好久，前一阵子的伦敦奥运会，微博首页右侧要添加一个奥运金牌榜的模块，要求有收起和展开的按钮，用来显示不同的内容。

展开状态

收起状态

对网站来说这是很稀松平常的交互效果。具体 html 实现可能有同学会想到，做两个 div，各自包含展开的内容和收起的内容。在点击展开按钮时出现一个，另一个隐藏；而在点击收起的时候做相反的处理。这种事本身也没有对与错，能实现效果就好。但作为出现在微博首页的模块，并且出现在第一屏的位置，对性能的优化肯定是要做足的。能尽量在我们 css 这一层做的，决不放到 js 那边去做。我的处理方式是把收起展开的样式都写好，放在一起，让 js 在默认展开或点击展开的时候显示 turn_olym_on，在点击收起的时候更换为 turn_olym_off，这样 js 就只是更换一个■

本文更详细内容，请查看：

http://developer.51cto.com/art/201210/362321.htm

2012 云计算架构师峰会视频采访报道专题

http://cloud.51cto.com/exp/WOT_interview/index.html

近几年，IT 技术与互联网的发展影响了整个产业的格局，带来了全新、鲜活的业务模式。面对这些变革，51CTO 作为中国领先的 IT 技术社区，站在技术发展的前沿，将于 10 月下旬举办架构师峰会，联合诸多 IT 技术的领导企业，共同搭建起一个架构师交流和沟通的平台，用以分享 IT 技术发展和应用的实践经验，交流国内外最新的技术动态和研究成果。届时，国内外知名技术专家、企业级 IT 技术与互联网创新型代表企业的同仁都将齐聚这里，共话 IT…

大会讲师 PPT 下载

http://wot.51cto.com/2012/download.html

大会现场直播专题

http://cloud.51cto.com/art/201211/363080.htm

大会现场直播专题 http://developer.51cto.com/exp/51ctofhzb/index.html